(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 466 389 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
07.10.1998 Bulletin 1998/41

(51) Int Cl.[6]: **G06F 11/14**, G06F 3/06

(21) Application number: **91306017.4**

(22) Date of filing: **02.07.1991**

(54) **File system with read/write memory and write once-read many (WORM) memory**

Dateisystem mit Schreib/Lesespeicher und einmaligen Schreib- und mehrmaligen Lese-speicher

Système de fichier comportant une mémoire lecture/écriture et une mémoire à écriture unique et lecture multiple

(84) Designated Contracting States:
DE FR GB IT

(30) Priority: **11.07.1990 US 551218**

(43) Date of publication of application:
15.01.1992 Bulletin 1992/03

(73) Proprietor: **AT&T Corp.**
New York, NY 10013-2412 (US)

(72) Inventor: **Thompson, Kenneth L.**
Watchung, New Jersey 07023 (US)

(74) Representative:
Watts, Christopher Malcolm Kelway, Dr. et al
Lucent Technologies (UK) Ltd,
5 Mornington Road
Woodford Green Essex IG8 OTU (GB)

(56) References cited:

- **THE JOURNAL OF SYSTEMS AND SOFTWARE vol. 10, no. 1, July 1989, USA pages 15 - 21 D. A. CANAS 'A File Management System for a Magnetic Disk Used as a Buffer to Write-Once Optical Storage'**
- **COMPUTER DESIGN vol. 28, no. 1, January 1989, TULSA, OK, USA pages 93 - 96 , XP000098294 R. B. OLSEN ET AL. 'Virtual optical disks solve the on-line storage crunch'**
- **INTERNATIONAL CONFERENCE ON DATA ENGINEERING 1984, SILVER SPRINGS, USA pages 175 - 180 P. RATHMANN 'Dynamic Data Structures on Optical Disks'**
- **7TH INT. CONF. ON DECISION SUPPORT SYSTEMS June 1987, SAN FRANCISCO, USA pages 107 - 113 G. DIEHR ET AL. 'USING OPTICAL STORAGE TECHNOLOGY FOR DECISION SUPPORT DATABASES'**
- **IEEE COMPUTER June 1988, USA pages 11 - 22 J. GAIT 'The Optical File Cabinet: A Random-Access File System for Write-Once Optical Disks'**

EP 0 466 389 B1

Printed by Jouve, 75001 PARIS (FR)

Description

Background of the Invention

1. Field of the Invention

The present invention is related to data storage in digital computers generally and more specifically to data storage using write once-read many (WORM) devices.

2. Description of the Prior Art

Various automated backup techniques have been developed for computer systems. Recently, automated backup systems have emerged in which the copies of the backed up files are stored on write once-read many or *WORM* devices. As the name implies, data can be written once to a WORM device and the data stored thereon can be read many times. Modern WORM devices are optical devices which provide random access to enormous quantities of data. A description of an optical disk WORM device may be found in

Gait, Jason, "The Optical File Cabinet: A Random-Access File System for Write-once Optical Disks", *IEEE Computer*, June, 1988, pp. 11-22

An example of a file backup system employing an optical disk can be found in

Hume, Andrew, "The File Motel - An Incremental Backup System for UNIX" *1988 Summer Usenix Conference Proceedings*, June 20-24, 1988, pp. 61-72

A problem which the above file backup system shares with many others is that the backup copies of the files are not as accessible to users of a computer system as the files presently on the system. In some cases, the backup copies must be physically retrieved from an archive and loaded onto the computer system; in others, like the system described in the above publication, the files are physically available but must be specifically mounted on the file system before they are accessible. Further, special tools are often required to deal with the backup files.

Of course, if a file system is stored on media which cannot be erased, then the need for backups to protect against human mistakes or malice or equipment failures is eliminated. The art has thus developed file systems in which all of the data is stored on an optical WORM system. One such file system is described in the Gait article cited above. While such file systems are essentially indestructible, they are not without their problems. First, since the entire file system is stored on optical disk, many disk blocks are wasted on the storage of transient files, i.e., files which are created and deleted in the course of execution of a program. Second, optical WORM devices are still substantially slower than magnetic disk devices, and file system performance suffers accordingly. While the speed problem can be alleviated by encaching data which has been read from the optical WORM device so that there is no need to retrieve it from the WORM device for a following read, encachement cannot solve the problem of wasted disk blocks. Further, though Gait's WORM file system contains substantially all of the data that was ever in the file system, it includes no provision for making backups at times that are significant to the users of the system, and therefore does not provide a way of reconstituting a file system exactly as it was at such a significant time.

In *The Journal of Systems and Software,* vol. 10, no. [1], July 1989, USA, pages 15-21, D.A. Cañas and W.D. Bulgren disclose a file management system for a magnetic disk used as a buffer to write-once optical storage. The data structures of the optical disk file management system permit the current state of stored files to be reconstructed from the information retained in the magnetic buffer and the previous version of the file permanently stored on the optical disk.

What is needed, and what is provided by the invention of claim 1, is a file system in which the user can select significant times to make backups and in which the backups made at these times are as available to the user as to any other files.

A method for dumping a primary file which is stored in such a file system is claimed in claim 4.

Further enhancements of the invention being provided by the subclaims.

Brief Description of the Drawing

FIG. 1 is an overview of the file system of the invention;

FIG. 2 is a diagram of a component file system of the invention;

FIG. 3 is a diagram of an operation which alters a primary file system in the file system of the present invention;

FIGS. 4A and 4B are diagrams of a dump operation in the file system of the present invention;

FIG. 5 is an overview of a preferred implementation of the present invention; and

FIG. 6 is an overview of a preferred implementation of the subdivision of mass storage device 507.

Reference numbers in the figures have two parts: the two right-most digits are numbers within a drawing; the left digit is the number of the drawing in which the item indicated by the reference number first appears. Thus, the item identified by the reference number 117 first appears in FIG. 1.

Detailed Description

The following Detailed Description of a preferred embodiment of the invention begins with a discussion of the logical structure of the file system of the invention, continues with a discussion of the operation of the file system of the invention, and concludes with a discussion of an implementation of the file system which em-

ploys an optical write once-read many optical disk device and a magnetic disk device.

Logical Structure of the File System: FIGS. 1 and 2

This discussion of the logical structure of the file system first provides an overview of the entire file system of the invention and then provides an overview of a component file system in the invention.

Overview of the File System: FIG. 1

FIG. 1 is a conceptual overview of file system 101 of the invention. All information contained in file system 101 is stored in storage elements (SE) 105. Each storage element 105 has a storage element address (SEA) 107 and is randomly accessible by storage element address 107. SE 105 may be implemented as a block on a randomly accessible device such as a magnetic or optical disk drive or a memory. The total number of possible storage element addresses 107, ranging from storage element address 107(0) through storage element address 107(max) makes up file address space (FAS) 103. The size of file address space 103 is, in principle, limited only by the size of storage element addresses 107; however, in some embodiments, its size may be determined by the size of the physical devices upon which storage elements 105 are stored.

Each storage element 105 belongs to one of three address spaces: read only address space 117, read/write address space 115, or unused address space 113. Storage elements 105 belonging to read-only address space 117 are inalterable components of file system 101; they may be read but neither written nor removed from file system 101. Storage elements 105 belonging to read-write address space 115 are alterable components of file system 101; they may be added to file system 101, written to, read from, and removed from file system 101. Storage element 105 belonging to unused address space 113, finally, are neither part of file system 101 nor presently available to be added to it.

At the beginning of operation of file system 101, all storage elements 105 belong to unused address space 113; when a storage element 105 is required for file system 101, file system 101 moves the storage element from unused address space 113 to read/write address space 115; when a storage element 105 has become an inalterable component of file system 101, file system 101 moves the storage element from read/write address space 115 to read-only address space 117. Once a storage element 105 is in read only address space 117, it remains there.

Consequently, as file system 101 operates, the number of storage elements 105 in unused address space 113 decreases and the number in read only address space 117 increases. When there are no more storage elements 105 in unused address space 113, the user must copy the files he needs from file system 101 onto another file system. File address space 103 can, however, be made so large that it is for practical purposes inexhaustible.

For the sake of simplicity, FIG. 1 presents the address spaces as though they were separated by clear boundaries in file address space 103. That is true only for unused address space 113. A storage element address HWM 108(c) marks the "high water mark" in file address space 103, i.e., the address of the first storage element which belongs to neither read/write address space 115 nor read only address space 117. All storage elements having addresses of HWM 108(c) or greater belong to unused address space 113. However, any storage element 105 having an address less than HWM 108(c) may belong either to read/write address space 115 or read only address space 117.

File address space 103 contains two kinds of component file systems: primary file system 111 and some number of dump file systems 109. Primary file system 111 behaves like a standard file system. Accordingly, all of the usual file operations may be performed on files in primary file system 111. Existing files may be read from, written to, and deleted; new files may be created. Files in dump file systems 109, on the other hand, may only be read. As is apparent from these properties, primary file system 111 may have storage elements 105 belonging to read/write address space 115 or read only address space 117, while all storage elements 105 of a dump file system 109 belong to read only address space 117. The line which appears in FIG. 1 at the top of each dump file system 109 represents the value of HWM 108 (c) at the time the dump file system 109 was created; the line is thus labeled with the number of dump file system 109. A number of dump file systems 109 may have the same value for HWM 108. Storage elements 105 belonging to a given component file system may be located anywhere in file address space 103 below HWM 108 for the file system, and a storage element 105 may be shared by more than one component file system.

A dump file system 109 is created by performing a dump operation on primary system 111. The dump operation has the logical effect of adding those storage elements 105 in read/write address space 115 which are part of primary file system 111 at the time of the dump operation to read only address space 117. The dump operation in file system 101 is atomic, i.e., no changes can be made in the files of primary system 111 during the dump operation. The dump operation accordingly conserves the state of primary file system 111 at the time the dump operation was performed. As a consequence of the manner in which the dump operation is performed, the dump file systems 109 are ordered in read only address space 117 by the time at which the dump operation which created the dump file system 109 was performed, with the dump file system 109 resulting from the earliest dump operation having the lowest HWM 108 in read only address space 117 and the dump file system 109 resulting from the most recent dump operation hav-

ing the highest HWM 108. The dump file systems 109 thus represent an ordered set of "snapshots" of past states of primary system 111.

Each component file system is organized as a tree, i.e., the storage elements 105 for all of the files in the component file system are accessible from a root 121 in the component file system. As previously indicated, storage elements 105 in primary file system 111 may belong to either read/write address space 115 or read only address space 117. The storage elements 105 belonging to read/write address space 115 are those which have new contents, i.e., those which contain parts of primary file system 111 which have been altered since the last dump operation. The storage elements 105 belonging to read only address space 117 are those which have old contents, i.e., those storage elements 105 in which portions of the file system are stored which have not been altered since the last dump operation.

As is apparent from the foregoing and the manner in which a dump file system 109 is created, the storage elements 105 of primary file system 111 which had new contents when dump file system 109 was created belong to the addition to read only address space 117 which was made when the dump operation was performed and the storage elements 105 of primary file system 111 which had old contents when dump file system 109 was created belong to the read only address space 117 which existed prior to the dump operation and are shared with earlier component file systems. This fact is indicated in FIG. 1 by shared element pointers (SEP) 129 in each component file system. The storage elements 105 pointed to by these pointers are shared with at least one other older component file system. In the case of the primary file system 111, such storage elements 105 are ones whose contents have not been altered since the last dump operation; in the case of a given dump file system 109, the shared element pointers 129 point to storage elements 105 which were not altered between the dump operation which created the preceding dump file system 109 and the dump operation which created the given dump file system 109. As is further apparent, a given storage element 105 in read only address space 117 is part of every component file system from the dump file system 109 resulting from the first dump operation after the given storage element 105 was incorporated into primary system 111 to the dump file system 109 (if any) produced by the dump operation immediately preceding the time at which the contents of the given storage element 105 were modified in the course of file operations on primary file system 111.

Each root 121 is itself accessible from location information block 119 in each component file system by means of root pointer (RP) 127, which points to storage element 105 which contains root 121. Additionally, each location information block 119 contains dump pointers (DPS) 125 to roots 121 in each component file system which precedes the component file system to which location information block 119 belongs and a next pointer (NP) 123 to the location information block 119 in the component file system which succeeds the component file system to which location information block 119 belongs. Location information block 119 for the first dump system 109(1), finally, is at a predetermined address in file address space 103. Every file in file system 101 may thus be located either directly from location information block 119(c) in primary file system 111 or indirectly from location information block 119(1). The chain of location information blocks beginning with location information block 119(1) is used to reconstruct location information block 119(c) in case of a failure of the physical device upon which read/write address space 115 is implemented. Location information block 119(c) serves in effect as a root for all of the files in file system 101, and it is consequently possible for a user of file system 101 to locate and read a file in a dump file system 109 in exactly the same way as the user would locate and read a file in primary file system 111. For example, from the user's point of view, comparing a version of a file in primary file system 111 with a version of the file in a dump file system 109 is no different from comparing two versions of the file in different directories of primary file system 111.

Detailed Structure of a Component File System: FIG. 2

FIG. 2 is a diagram of the structure of a component file system in file system 101. All of the information which is contained in the files and which is needed to organize the files into a file system is stored in storage elements 105. All of the component file systems have similar structures; however, in dump file systems 109, all of the storage elements 105 in the file system belong to read only address space 117, while primary file system 111 has some storage elements 105 which belong to read only address space 117 and others which belong to read/write address space 115.

The files in the component file systems are hierarchical. Each file belongs to a directory and a directory may contain files or other directories. The hierarchy has the form of a tree with a single root node. The directories are internal nodes of the tree and the files are the leaf nodes. In a preferred embodiment, there is a single path through the tree from the root to each leaf node, i.e., no file or directory belongs to more than one directory.

As shown in FIG. 2, component file system 201 has two main parts: location information 119 and file tree 202. Beginning with file tree 202, tree 202 has two kinds of elements: directory blocks (DB) 219, which represent directories, and data blocks (DATA) 225, which contain the data for a file. Directory blocks 219 contain two kinds of entries: file entries (FE) 221, which represent files belonging to the directory, and directory entries (DE) 223, which represent directories belonging to the directory. There is one file entry 221 and one directory entry 223 for each file and directory belonging to the directory. A file entry 221 contains data pointers (DATA PTRS) 229

to the data blocks 225 which contain the file's data; a directory entry 223 contains a directory pointer (DIR PTR) 231 to directory block 219 for the directory represented by directory entry 223. As will be explained in more detail below, data blocks 225 and directory blocks 219 may be shared with other older component file systems; when that is the case, the data pointers 229 to those data blocks and the directory pointers 231 to those directories are shared element pointers 129.

Location information 119 has three components in a preferred embodiment: superblock (SB) 203, dump list (DL) 211, and free list (FRL) 207. Superblock 203 contains pointers by which the other parts of a component file system can be located, is pointed to by next pointer 123 belonging to the preceding component file system, and itself contains next pointer 123 to superblock 203 for the following component file system. In the case of superblock 203 for primary file system 111, the superblock contains HWM 108(c). These contents are arranged in superblock 203 as follows: HWM 108 contains HWM 108(c) in primary file system 111 and the value of HWM 108(c) at the time the dump operation was performed in dump file systems 109. NP 123 contains next pointer 123 in dump file systems 109; RP 127 is the pointer to root 121 for the component file system; DLP 27 is a pointer to dump list 211; FRLP 205 is a pointer to free list 207.

Dump list 211 contains a list of all of the dump file systems 109 which precede the component file system to which dump list 211 belongs. Each entry (DLE) 213 in dump list 211 has two parts: dump identifier 215 and dump pointer 217. Dump identifier 215 is a unique identifier which identifies the dump file system 109 represented by dump list entry 213; in a preferred embodiment, dump identifier 215 specifies the time and date at which the dump operation which created dump file system 109 was carried out. Dump pointer 217 is a pointer to root 121 for the dump file system 109 represented by dump list entry 213. Taken together, the dump pointers in dump list 211 thus make up dump pointers (DPS) 125.

Free list 207, finally, is a list of addresses 107 of storage elements 105 which are no longer part of unused address space 113 but are not presently part of primary file system 111. For example, if a new file is created in primary file system 111 after the last dump operation and then deleted before the next dump operation, the addresses 107 of the storage elements 105 from the deleted file are placed on free list 207. Free list 207 is an important advantage of file system 101, since it permits the set of storage elements 105 belonging to read/write address space 115 to fluctuate between dump operations without a corresponding fluctuation of unused address space 113.

Though free list 207 is part of every component file system, it has significance only in primary file system 111, where it is the source of storage elements 105 to be added to primary file system 111, and the most recent dump file system 109, where it is used to reconstitute primary file system 111's free list 207 after a destruction of primary file system 111. When free list 207 in primary file system 111 becomes empty as a result of incorporation of free storage elements 105 into primary file system 111, file system 101 obtains a new storage element 105 from unused address space 113 by adding the current value of HWM 108(c) to free list 207 and incrementing HWM 108 in superblock 203.

As previously mentioned, the storage elements 105 making up primary file system 111 belong to either read-write address space 115 or read only address space 117. In more detail, the storage elements 105 containing the components of location information 119 and root 121 always belong to read/write address space 115, as do the storage elements 105 on free list 207. The parts of tree 202 are in read/write address space 115 as follows: Any directory block 219 which is part of a path to a file which is presently open for an operation which alters the file is contained in a storage element 105 in read/write address space 115; Any data block 225 which has been written to since the last dump operation is contained in a storage element 105 in read/write address space 115.

The technique by which storage elements 105 containing new contents replace those with shared contents will be explained in detail below.

Performing Operations on Component File Systems: FIG. 3

Operations on file systems can be divided into two classes: those which alter the file system and those which do not. Operations of the second class, termed hereinafter read operations, can be performed in the usual manner on any component file system of file system 101. Operations of the first class, termed hereinafter write operations, may be performed only on files and directories in primary file system 111. FIG. 3 shows how two of the write operations, file open and file write, are performed in a simple example primary file system 111 which contains exactly one file. Each block in FIG. 3 contains a number in parenthesis indicating the type of component represented by the block and an indication of the address space to which the storage element 105 containing the component belongs. Thus, root 121 is a directory block 219 and belongs to read/write address space 115.

The portion of FIG. 3 labeled 301 shows the example primary file system 111 at a time after the last dump operation but before the single file has been opened. Only root 121 belongs to read/write address space 115; the remaining components, including directory 303 to which the file belongs and the data blocks 305 and 307 for the file belong to read only address space 117, i.e., directory 303 and data blocks 305 and 307 are shared at least with dump file system 109 made by the last dump operation, as indicated by pointer 302 from root 121 for the immediately preceding dump file system 109.

The next portion, labeled 309, shows the example primary file system 111 at a time after the single file has been opened for writing but before any file write operation has occurred. Because the file has been opened, it must have a directory block 219 in read/write address space 115. This directory block 219, which has the number 311 in the FIG., is made by taking a storage element 105 from free list 207, copying the contents of directory block 303 into directory block 311, and changing pointer 231 in root 121 to point to directory block 311 instead of directory block 303. Data blocks 305 and 307 are now pointed to by both directory block 303 and directory block 311, as indicated by 312 and 304, representing data pointers 229 pointing to the data blocks.

The final portion, labeled 313, shows the example primary file system 111 after a file write operation which has altered data originally contained in data block 307. The altered data requires a new data block, block 315, which belongs to the read/write address space 115. Block 315 is taken from free list 207 as before and the data pointers 229 in directory block 311 are reset so that block 315 takes the place of block 307. Then the altered data is written to block 315. At the end of the operation, file entry 221 for the file in directory 311 points to blocks 305 and 315, as indicated by pointer 317, while file entry 221 for the file in directory 303 still points to blocks 305 and 307 as indicated by pointer 304. Thus, the original file is retained in dump file system 109 and the altered file in primary system 111.

Other standard file operations are performed analogously. For example, when a file is created, a new file entry 221 for the file is made in a directory block 219; if the directory block 219 already belongs to read/write address space 115, the new file entry is simply added to the directory block 219; otherwise a new directory block 219 is made as described above for the open operation and the new file entry 221 is added to the new directory block. In the case of a file delete operation, only those data blocks of the file to be deleted which belong to read/write address space 115 can be deleted; this is done by returning the addresses of storage elements 105 containing the deleted data blocks 225 to free list 207. At the same time, the file entry 221 in the directory block 219 for the directory to which the file belongs is also deleted. If all of the file entries 221 and directory entries 223 in a directory block 219 are deleted, that block's storage element 105, too, is returned to free list 207. As shown by the delete example, a particular advantage of file system 101 is that changes in primary file system 111 which affect primary file system 111 for a period which is less than the period between dump operations take place in read/write address space 115 and do not add storage elements 105 to read only address space 117.

The Dump Operation: FIGS. 4A and 4B

In broad terms, the dump operation creates a dump file system 109 by moving storage elements 105 in primary file system 111 which belong to read/write address space 115 from read/write address space 115 to read only address space 117 and reestablishing the parts of primary file system 111 which must be presently alterable in read/write address space 115. In a preferred embodiment, the algorithm for performing the dump operation is the following: Set next pointer 123 in superblock 203 for primary file system 111 to HWM 108; Add all storage elements 105 for primary file system 111 which belong to read/write address space 115 and which are not on free list 207 to read only address space 117; Reestablish primary file system 111 in read/write address space 115 by doing the following: Copy the contents of superblock 203 to storage element 105 having HWM 108 as its storage element address 107 to make a new superblock 203 for the primary file system 111 and increment HWM 108 in the new superblock 203 to point to the next storage element 105; Taking storage elements 105 from free list 207, copy location information 119 and root 121 to the storage elements 105; Update the pointers in the new superblock 203 to point to the location information 119 and root 121 in the storage elements 105 taken from free list 207; Add an entry for the new dump file system 109 to dump list 211; and For every file in primary file system 111 which is open at the time of the dump operation, walk tree 202 from root 121 in new read/write address space 115 to directory block 219 which contains file entry 221 for the file; for each directory block 219 encountered in the walk which is not yet in new read/write address space 115, copy the directory block 219 to a storage element 105 taken from free list 207 and alter directory entry 223 and any copies of information from directory entry 223 in the computer system to which file system 101 belongs to point to the new copy.

The algorithm is performed atomically, i.e., no changes to file system 101 other than those required for the algorithm are permitted during execution of the algorithm.

In an alternative embodiment, the contents of superblock 203 may be copied to a new superblock 203 taken from free list 207, next pointer 123 updated to point to the new superblock 203, and then all storage elements 105 in read/write address space 115 other than the new superblock 203 added to read only address space 117.

FIG. 4 shows how the dump operation works in a primary file system 111 which contains exactly one directory in which there is exactly one closed file. Again, each box in the file contains a reference number indicating what kind of component of primary file system 111 the box represents and an indication of which of the address spaces 115 and 117 the component belongs to part 402 of FIG. 4 shows primary file system 401 as it exists at the time of the dump: Components 401, 403, and 405, making up location information 119, root 407 and directory block 409 all belong to read/write address

space 115; the file has one data block 411 which has not been altered since the last dump operation, and consequently belongs to read only address space 117; the other data block 413 has been altered, and so belongs to read/write address space 115.

Part 415 of FIG. 4 shows primary file system 415 as it exists after completion of the dump: components 401-409 and 413 have all been moved to read-only address space 117, copies 417-423 in read/write address space 115 have been made of components 401-407, pointers in the copies of location information 119 have been set so that root pointer 127 points to new root 423 and dump pointer 217 points to old root 407, and next pointer 123 in old super block 401 has been set to point to new superblock 417. If the file had been open, there would additionally have been a copy of directory block 409, and new root 423 would point to the copy.

Replacing Primary System 111 With a Dump File System 109

An advantage of file system 101 is that primary file system 111 may be easily replaced by any of the dump file systems 109. Replacement is done by performing the following steps, again atomically: Except for those which contain location information 119, return the addresses of all storage elements 105 in primary file system 111 which belong to read/write space 115 to free list 207; Using dump pointer 217 for the dump file system 109 which is replacing primary file system 111, locate root 121 for the dump file system 109 and copy root 121 into a storage element 105 from free list 207; Replace root pointer 127 in superblock (SB) 203 with a pointer to the copy of root 121 for the dump file system 109.

If a failure in the computer system to which the file system belongs has resulted in the loss of location information 119, the location information 119 may be copied from location information 119 in the most recent dump file 109. In this case, if the dump file system 109 which is replacing primary file system 111 is the most recent dump file system 109, then the second step above is omitted.

Implementation of File System 101 on a Read/write Mass Storage Device and a WORM Storage Device: FIGS. 5 and 6

In a preferred embodiment, file system 101 is implemented using a read-write mass storage device and a WORM storage device. In the following, there will first be presented an overview of the implementation and the relationship of its components to file address space 103; then details of the organization of the read/write mass storage device will be presented, followed by details concerning the operation of the preferred embodiment.

Overview of the Implementation: FIG. 5

FIG. 5 is a high-level block diagram of a preferred implementation of file system 101. Implementation 501 has three main components: file server 503, random access read/write mass storage device 507, and random access write once-read many (WORM) device 511. File server 503 is a computer system which is employed in a distributed system to perform file operations for other components of the distributed system. In the preferred implementation, file server 503 is a VAX 750, manufactured by Digital Equipment Corporation. The file operations which it performs for other components are substantially the same as those defined for the well-known UNIX® operating system. File server 503 controls operation of the other components of implementation 501. Mass storage device 507 in a preferred embodiment is 120 megabytes of storage on a magnetic disk drive. WORM device 511 is the WDD-2000, a 1.5 gigabyte write-once optical disk manufactured by Sony, Inc.

The storage in both mass storage device 507 and WORM device 511 is divided into blocks of the same size. These blocks make up the storage elements 105 of the implementation. The blocks in mass storage device 507 appear in FIG. 6 as disk blocks (DB) 509 and those in WORM device 511 appear as WORM blocks (WB) 519. As will be explained in more detail later, disk blocks 509 correspond to certain WORM blocks 519. Such correspondences are indicated by letter suffix. Thus, disk block 509(a) corresponds to WORM block 519(a). File server 503 may both read and write disk blocks 509 many times; it may read WORM blocks 519 many times, but write them only once. These facts are indicated in FIG. 5 by read/write operations arrow 505 connecting file server 503 and mass storage device 507 and the separate write once operation arrow 513 and read operation arrow 515 connecting file server 503 and WORM device 511. They are further indicated by the division of WORM device 511 into unwritten portion 515 containing WORM blocks 519 which have not yet been written and written portion 517, containing written WORM blocks 519. Since WORM device 511 is random access, written and unwritten blocks may be physically intermixed.

Before a dump operation, the relationship between disk blocks 509 and WORM blocks 519 and file address space 103 is the following: Written WORM blocks 519 belong to read-only address space 117, as do corresponding disk blocks 509; such disk blocks 509 contain copies of the contents of the corresponding WORM blocks 519; Disk blocks 509 which correspond to unwritten WORM blocks 519 belong to read/write address space 115, as do the corresponding unwritten WORM blocks 519, which are reserved to receive the contents of the corresponding disk blocks 509 after a dump operation is performed, as indicated by the label "dump space 516" in their portion of WORM device 511; Unwritten WORM blocks 519 which have no corresponding

disk blocks 509 belong to unused address space 113.

As implied by the above, disk blocks 509 which correspond to written WORM blocks 519 and those which correspond to unwritten WORM blocks 519 have fundamentally different functions in file system 101. That is shown in FIG. 5 by the division of mass storage device 507 into two parts: read only cache 506 and read/write store 508. Again, since mass storage device 507 is a random-access device, disk blocks 509 belonging to either subdivision may be located anywhere in mass storage device 507.

Disk blocks 509 corresponding to written WORM blocks 519 serve as a cache 506 of those blocks. Mass storage device 507 has a faster response time than WORM device 511, and consequently, when file server 503 reads a written WORM block 519 from WORM device 511, it places a copy of that WORM block 519 in a disk block 509 so that it is available there if it is needed again. As is generally the case with caches, cache 506 contains copies of only a relatively small number of the most recently read WORM blocks 519. While cache 506 substantially enhances performance, it is not necessary to an implementation of file system 101.

Disk blocks 509 corresponding to unwritten WORM blocks 519, on the other hand, are the actual storage elements 105 for read/write address space 115 and are therefore essential to operation of file system 101. There must be a disk block 509 in read/write store 508 for every storage element 105 which is a part of primary file system 111. Since dump space 516 is reserved to receive the contents of read/write address space 115 when a dump operation is performed, there must be a WORM block 519 in dump space 516 corresponding to every disk block 509 in read/write store 508. Additionally, there must be a WORM block 519 in dump space 516 corresponding to every storage element 105 on free list 207. The storage elements 105 on free list 207, however, need not have corresponding disk blocks 509 belonging to read/write store 508. Accordingly, when free list 207 becomes empty and a storage element 105 must be added to read/write address space 115, dump space 516 must be expanded by one unwritten WORM block 519. As indicated by the presence of HWM 108(c) at the end of dump space 516, that is done by incrementing HWM 108(c).

For a time after completion of a dump operation, mass storage device 507 contains a third subdivision: dump store 510. Dump store 510 contains disk blocks 509 which are storage elements 105 which have been added to read only address space 117 by the dump operation. A disk block 509 remains in dump store 510 until its contents have been copied into the corresponding WORM block 519 in dump space 516. Upon being written, the WORM block 519 becomes part of read-only address space 117 and its corresponding disk block 509 becomes part of read-only cache 506.

As will be explained in more detail later, the actual dump operation in a preferred implementation simply marks disk blocks 509 which contain parts of primary file system 111 which are in read/write address space 115 as belonging to dump store 510. As soon as this is done, normal file operations continue. These operations treat components of primary file system 111 which are stored in disk blocks 509 belonging to dump store 510 as part of read-only address space 117. While these operations are going on, a dump daemon which executes independently in file server 503 copies the contents of the disk blocks 509 to the corresponding WORM blocks 519 in dump space 516. Once a disk block 509 has been copied, it is marked as belonging to read-only cache 506.

It should be pointed out at this point that file address space 103 may be larger than the address space of WORM device 511. To begin with, unused address space 113 may extend beyond the top address in WORM device 511. Further, as long as all storage elements 105 belonging to the component file systems being operated on by file system 101 are on WORM device 511, read-only address space 117 can extend below the lowest address in WORM device 511.

Implementation of the Subdivisions of Mass Storage Device 507: FIG. 6

In a preferred implementation, correspondences between disk blocks 509 and WORM blocks 519 and division of mass storage device 507 into read only cache 506, read/write store 508, and dump store 510 are established by means of a mass storage map 603, shown in FIG. 6. In the preferred implementation, mass storage map 603 is a data structure in virtual memory 611 of file server 503. Since mass storage map 603 is used in every file operation performed by file system 101, it is generally present in the main memory of file server 503 and therefore rapidly accessible to file server 503.

Map 603 is an array of map entries 605. There is a map entry 605 for each disk block 509 which is part of mass storage device 507, and the address of the disk block 509 represented by a given map entry 605 may be calculated from the index of the given map entry 605 in the array, as is indicated by the arrows connecting the first and last map entries 605 in map 603 to the first and last disk blocks 509 in mass storage device 507.

Moreover, the number of WORM blocks 519 in WORM device 511 is an integer multiple m of the number b of disk blocks 509 and map entries 605. Consequently, the index (I) of a map entry 605 may be computed from a WORM block address (WBA) by the operation WBA MOD m, and the disk block 509 represented by a given map entry 605 may correspond to any WORM block 519 for which I = WBA MOD m, where I is the index of the given map entry 605.

Each map entry 605 contains two fields. The first, storage element address field 607, contains storage element address 107 representing WORM block 519 corresponding to disk block 509 represented by map entry

605. In a preferred implementation, storage element address 107 is simply a WORM block address. The second field, disk block state 609, indicates the present state of disk block 509 represented by map entry 605. There are four states: Not bound: There is no WORM block 519 corresponding to disk block 509; Read only: Disk block 509 corresponds to the WORM block 519 specified by field 607. The WORM block is a storage element 105 belonging to read only address space 117, disk block 509 contains a copy of WORM block 519's contents, and disk block 509 belongs to read only cache 506. Read/write: Disk block 509 corresponds to the WORM block 519 specified by field 607. The WORM block 519 belongs to dump space 516, disk block 509 is a storage element 105 of primary file system 111 which is in read/write address space 115, and disk block 509 belongs to read/write store 508. Dump: Disk block 509 corresponds to the WORM block 519 specified by field 607. The WORM block 519 belongs to dump space 516, disk block 509 represents a storage element 105 in read only address space 117, and disk block 509 belongs to dump store 510.

Implementation 501 operates as follows: when a file read operation is performed, file server 503 uses the storage element address 107 of the storage element 105 to be read to locate a map entry 605. That address is termed herein address "A". What then happens depends on whether there is a disk block 509 corresponding to WORM block 519 addressed by A in mass storage device 507. If there is, field 607 in map entry 605 located by address A will contain address A. If it does and the map entry 605 is in the read only, read/write, or dump states, disk block 509 represented by map entry 605 contains the desired data and file server 503 reads the contents of disk block 509.

If there is not a disk block 509 corresponding to WORM block 519 in mass storage device 507 but the map entry 605 indicates that the disk block 509 it represents has the read only state, file server 503 copies the contents of WORM block 519 specified by address A into disk block 509 corresponding to map entry 605 and writes address A into field 607. If map entry 605 indicates the read/write or dump state, file server 503 cannot overwrite the contents of disk block 509 corresponding to map entry 605 and simply fetches the data from WORM block 519. If map entry 605 indicates the not bound state, finally, file server 503 fetches the data from WORM block 519, copies it into disk block 509 corresponding to map entry 605, writes address A into field 607, and sets DBS 609 to indicate the read only state.

When an operation which alters a file is performed, file server 503 again uses storage element address 107 A to locate a map entry 605. If field 607 in map entry 605 contains A and indicates the read only or dump states, file server 503 takes a storage element address 107 B from free list 207 and uses address B to locate a second map entry 605. If this map entry 605 is in the not bound or read only states, file server 503 copies the contents of the disk block 509 represented by the map entry 605 addressed by A to the disk block 509 represented by the map entry 605 addressed by B, sets field 607 in that map entry 605 to address B, and field 609 to indicate the read/write state. Pointers are updated in the file structure as already described and alterations are made to the disk block 509 represented by map entry 605 addressed by B. If the second map entry 605 is in the dump or read/write states, file server 503 takes another address from free list 207 and tries again.

If address A is different from the address in field 607, then if map entry 605 indicates the read only state, file server 503 copies the WORM block 519 addressed by A into the disk block 509 represented by entry 605, sets the fields of map entry 605 accordingly, and proceeds as described above for map entries 605 in the read only state. If map entry 605 indicates the read/write state or dump state, file server 503 immediately copies the contents of disk block 509 represented by map entry 605 to the corresponding WORM block 519, which places disk block 509 in the read only state, and then proceeds as just described for map entries 605 indicating the read only state.

When file server 503 deletes a file, it locates map entry 605 for each disk block 509 which is a data block 225 in the file. If there is no entry, or if the map entry 605 indicates the read only or dump states, the file server 503 does nothing; if the map entry 605 indicates the read/write state, the file server 503 adds the storage element address 107 of the corresponding WORM block 519 to free list 207 and places the map entry 605 for the disk block 509 into the not bound state.

To begin a dump operation, file server 503 goes through map 603 and places all map entries 605 which are in the read/write state in the dump state. It then reestablishes primary file system 111 as previously described using the file operations just described. At this point, the dump operation is finished. In a preferred embodiment, the operation takes about 10 seconds. During that time, file system 101 is unavailable for use. While file server 503 continues normal file operations, a dump daemon process operating in file server 503 writes the contents of all disk blocks 509 represented by map entries 605 indicating the dump state to their corresponding WORM blocks 519. When the write of a disk block 509 is completed, DBS field 609 is set to indicate the read only state.

In an alternative embodiment, there may be an additional state, "old superblock" for disk blocks 509. In such an embodiment, when a dump operation begins, the disk block 509 containing superblock 203 for primary file system 111 is placed in the "old superblock" state until the address of disk block 509 for the new superblock 203 has been taken from free list 207 and copied into superblock 203. At that point, disk block 509 containing superblock 203 is placed in the dump state.

Conclusion

The foregoing Detailed Description has disclosed to one of ordinary skill in the arts to which the invention pertains how file system 101 of the invention may be implemented. While a preferred embodiment of file system 101 is implemented using a magnetic disk drive and a WORM optical disk drive, file system 101 may be implemented using any devices which provide random-access read many-write many storage and random-access write once-read many storage.

Claims

1. A file system for use in a computer system having read-write (509) and write once/read many (519) storage elements for storing file contents, the file system comprising:

   at least one primary file (111) whose contents include old contents (303, 305, 307) which are stored in written ones of the write once/read many storage elements and new contents (311, 315) which are stored in the read-write elements and

   means for performing file operations (503) the file operations including at least a write operation and a separate dump operation, the write operation operating to write the new contents to the read-write storage elements wherein a correspondence between the read-write elements in which new contents are stored and unwritten write once-read many elements exists at the time the new contents are stored and the dump operation operating to write the new contents of a given read write element to corresponding unwritten ones of the write once/read many storage elements and the new contents being written to the unwritten ones of the write once/read many storage elements only in response to a dump command, and at no other time, the dump operation operates to reestablish the primary file such that the reestablished primary file contains only old contents including the new contents being written to unwritten ones of the once-write/read many storage elements by the dump operation, and

   the dump operation having

   a first part in which the read-write elements in which new contents are stored are atomatically marked as having been dumped; and

   a second part in which the marked read-write elements are nonatomatically written to their corresponding unwritten write once-read many elements (516).

2. The file system set forth in claim 1 wherein:

   all contents of the primary file which have not been altered since the last dump operation are old contents and all contents of the primary file which have been added or altered since the last dump operation are new contents.

3. The file system set forth in claim 2 wherein:

   the file operations further include read operations which read the contents of a file;

   the dump operation further operates to create a secondary file which conserves the state of the primary file at the time of the dump operation and whose contents are stored in the write once/read many storage elements written by the dump operation and in the write once/read many storage elements which contained the old contents at the time the dump operation is performed; and

   the read operations only may be performed on the secondary file using an interface which is the same as that used for read operations on the primary file.

4. A method for dumping a primary file which is stored in a file system having read-write (509) and write once/read many (519) storage elements for storing file contents, the file's contents including old contents (303, 305, 307) which are stored in written ones of the write once/read many storage elements and new contents (311, 315) which are stored in the read-write elements, the method comprising the steps of:

   as part of the operation of storing new contents in a given read-write storage element, establishing a correspondence between the given read-write storage element and a given unwritten write once/read many storage element; and

   in response to a dump command, and at no other time, copying the contents of the read-write storage elements to the corresponding unwritten write once/read many storage elements (516),

   the step of copying the new contents to the corresponding unwritten write once/read many storage elements includes the steps of

   atomatically marking all of the read-write storage elements which contain the new contents; and

   nonatomically copying the contents of the marked read-write storage elements to their corresponding unwritten write once/read many storage elements.

5. The method set forth in claim 4 and further comprising the step of:

   creating a secondary file which conserves the

state of the primary file at the time of the dump command and whose contents are stored in the write once/read many storage elements written in response to the dump command and in the write once/read many storage elements containing the old contents at the time of the dump command.

6. The method set forth in claim 5 wherein:

predetermined old contents of the primary file are additionally stored in the read-write storage elements;
the step of creating the secondary file further includes the step of further atomically marking all of the read-write storage elements which contain the new contents to indicate that the read-write storage elements contain old contents.

Patentansprüche

1. Dateisystem zur Verwendung in einem Computersystem mit Schreib-/Lese- (509) und einmal beschreibbaren und mehrfach lesbaren (519) Speicherelementen zur Speicherung von Dateiinhalten, wobei das Dateisystem folgendes umfaßt:

mindestens eine Primärdatei (111), deren Inhalte alte Inhalte (303, 305, 307), die in beschriebenen der einmal beschreibbaren und mehrfach lesbaren Speicherelemente gespeichert sind, und neue Inhalte (311, 315), die in den Schreib-/Lese-Elementen gespeichert sind, enthalten, und
Mittel zur Durchführung von Dateioperationen (503) zu denen mindestens eine Schreiboperation und eine separate Auszugsoperation gehören, wobei die Schreiboperation wirkt, um die neuen Inhalte in die Schreib-/Lese-Speicherelemente zu schreiben, wobei zum Zeitpunkt der Speicherung neuer Inhalte eine Entsprechung zwischen den Schreib-/Lese-Elementen, in denen die neuen Inhalte gespeichert werden, und unbeschriebenen einmal beschreibbaren und mehrfach lesbaren Speicherelementen besteht, und die Auszugsoperation wirkt, um die neuen Inhalte eines gegebenen Schreib-/Lese-Elements in entsprechende unbeschriebene der einmal beschreibbaren und mehrfach lesbaren Speicherelemente zu schreiben, und wobei die neuen Inhalte nur als Reaktion auf einen Auszugsbefehl und zu keinem anderen Zeitpunkt in die unbeschriebenen der einmal beschreibbaren und mehrfach lesbaren Speicherelemente geschrieben werden, wobei die Auszugsoperation wirkt, um die Primärdatei wiederherstellt, so daß die wiederhergestellte Primärdatei nur alte Inhalte, einschließlich der neuen Inhalte, die durch die Auszugsoperation in unbeschriebenen der einmal beschreibbaren und mehrfach lesbaren Speicherelemente geschrieben werden, enthält, und wobei
die Auszugsoperation folgendes aufweist:
einen ersten Teil, bei dem die Schreib-/Lese-Elemente, in denen neue Inhalte gespeichert sind, atomatisch als ausgezogen markiert werden; und
einen zweiten Teil, bei dem die markierten Schreib-/Lese-Elemente nichtatomatisch in ihre entsprechenden unbeschriebenen einmal beschreibbaren und mehrfach lesbaren Speicherelemente (516) geschrieben werden.

2. Dateisystem nach Anspruch 1, wobei:
alle Inhalte der Primärdatei, die seit der letzten Auszugsoperation nicht verändert wurden, alte Inhalte sind, und alle Inhalte der Primärdatei, die seit der letzten Auszugsoperation hinzugefügt oder verändert wurden, neue Inhalte sind.

3. Dateisystem nach Anspruch 2, wobei:

zu den Dateioperationen weiterhin Leseoperationen gehören, die Inhalte einer Datei lesen;
die Auszugsoperation weiterhin wirkt, um eine Sekundärdatei zu erzeugen, die den Zustand der Primärdatei zum Zeitpunkt der Auszugsoperation bewahrt und deren Inhalte in den durch die Auszugsoperation beschriebenen einmal beschreibbaren und mehrfach lesbaren Speicherelementen und in den einmal beschreibbaren und mehrfach lesbaren Speicherelementen, die die alten Inhalte zum Zeitpunkt der Ausführung der Auszugsoperation enthielten, gespeichert sind; und
an der Sekundärdatei nur die Leseoperationen ausgeführt werden dürfen, wobei eine Schnittstelle verwendet wird, die mit der für Leseoperationen an der Primärdatei verwendeten Schnittstelle übereinstimmt.

4. Verfahren für den Auszug einer Primärdatei, die in einem Dateisystem mit Schreib-/Lese- (509) und einmal beschreibbaren und mehrfach lesbaren (519) Speicherelementen zur Speicherung von Dateiinhalten gespeichert ist, wobei die Inhalte der Datei alte Inhalte (303, 305, 307), die in beschriebenen der einmal beschreibbaren und mehrfach lesbaren Speicherelemente gespeichert sind, und neue Inhalte (311, 315), die in den Schreib-/Lese-Elementen gespeichert sind, enthalten, wobei das Verfahren die folgenden Schritte umfaßt:

als Teil der Operation der Speicherung neuer

Inhalte in einem gegebenen Schreib-/Lese-Speicherelement, Herstellung einer Entsprechung zwischen dem gegebenen Schreib-/Lese-Speicherelement und einem gegebenen unbeschriebenen einmal beschreibbaren und mehrfach lesbaren Speicherelement; und
als Reaktion auf einen Auszugsbefehl und zu keinem anderen Zeitpunkt, Kopieren der Inhalte der Schreib-/Lese-Speicherelemente in die entsprechenden unbeschriebenen einmal beschreibbaren und mehrfach lesbaren Speicherelemente (516),
wobei der Schritt des Kopierens der neuen Inhalte in die entsprechenden unbeschriebenen einmal beschreibbaren und mehrfach lesbaren Speicherelemente die folgenden Schritte umfaßt:
atomatisches Markieren aller Schreib-/Lese-Speicherelemente, die die neuen Inhalte enthalten; und
nichtatomisches Kopieren der Inhalte der markierten Schreib-/Lese-Speicherelemente in ihre entsprechenden unbeschriebenen einmal beschreibbaren und mehrfach lesbaren Speicherelemente.

5. Verfahren nach Anspruch 4, weiterhin mit dem folgenden Schritt:

Erzeugen einer Sekundärdatei, die den Zustand der Primärdatei zum Zeitpunkt des Auszugsbefehls bewahrt und deren Inhalte in den einmal beschreibbaren und mehrfach lesbaren Speicherelementen, die als Reaktion auf den Auszugsbefehl beschrieben werden, und in den einmal beschreibbaren und mehrfach lesbaren Speicherelementen, die die alten Inhalte zum Zeitpunkt des Auszugsbefehls enthalten, gespeichert sind.

6. Verfahren nach Anspruch 5, wobei:

vorbestimmte alte Inhalte der Primärdatei zusätzlich in den Schreib-/Lese-Speicherelementen gespeichert werden;
der Schritt des Erzeugens der Sekundärdatei weiterhin den Schritt des weiteren atomischen Markierens aller Schreib-/Lese-Speicherelemente, die die neuen Inhalte enthalten, umfaßt, um anzuzeigen, daß die Schreib-/Lese-Speicherelemente alte Inhalte enthalten.

Revendications

1. Système de fichier destiné à être utilisé dans un système d'ordinateur ayant des éléments de mémoire à lecture-écriture (509) et à écriture unique-lecture multiple pour mémoriser un contenu de fichier, le système de fichier comprenant :

au moins un fichier principal (111) dont le contenu comporte un ancien contenu (303, 305, 307) qui est mémorisé dans des éléments de mémoire à écriture unique-lecture multiple écrits et un nouveau contenu (311, 315) qui est mémorisé dans les éléments à lecture-écriture et
un moyen pour effectuer des opérations de fichier (503), les opérations de fichier comportant au moins une opération d'écriture et une opération de vidage séparée, l'opération d'écriture fonctionnant pour écrire le contenu dans les éléments de mémoire à lecture-écriture dans lesquels une correspondance entre les éléments à écriture-lecture dans lesquels le nouveau contenu est mémorisé et des éléments à écriture unique-lecture multiple non écrits existe au moment où le nouveau contenu est mémorisé et l'opération de vidage fonctionnant pour écrire le nouveau contenu d'un élément à lecture-écriture donné dans des éléments de mémoire à écriture unique-lecture multiple non écrits correspondants et le nouveau contenu étant écrit dans les éléments de mémoire à écriture unique-lecture multiple non écrits uniquement en réponse à une commande de vidage, et à aucun autre moment, l'opération de vidage fonctionne pour rétablir le fichier principal de telle sorte que le fichier principal rétabli ne contienne qu'un ancien contenu y compris le nouveau contenu écrit dans les éléments de mémoire à écriture unique-lecture multiple non écrits par l'opération de vidage, et l'opération de vidage ayant
une première partie dans laquelle les éléments de lecture-écriture dans lesquels le nouveau contenu est mémorisé automatiquement sont marqués comme ayant été vidés ; et
une deuxième partie dans laquelle les éléments à lecture-écriture marqués sont écrits non automatiquement dans leurs éléments à écriture unique-lecture multiple non écrits correspondants (516).

2. Système de fichier selon la revendication 1, dans lequel :

tout le contenu du fichier principal qui n'a pas été modifié depuis la dernière opération de vidage est un ancien contenu et tout le contenu du fichier principal qui a été ajouté ou modifié depuis la dernière opération de vidage est un nouveau contenu.

3. Système de fichier selon la revendication 2, dans lequel :

les opérations de fichier comportent en outre des opérations de lecture qui lisent le contenu d'un fichier ;

l'opération de vidage fonctionne en outre pour créer un fichier secondaire qui conserve l'état du fichier principal au moment de l'opération de vidage et dont le contenu est mémorisé dans les éléments de mémoire à écriture unique-lecture multiple écrits par l'opération de vidage et dans les éléments de mémoire à écriture unique-lecture multiple qui contenaient l'ancien contenu au moment de l'exécution de l'opération de vidage ; et
les opérations de lecture ne peuvent être effectuées sur le fichier secondaire qu'en utilisant une interface qui est la même que celle utilisée pour les opérations de lecture sur le fichier principal.

4. Procédé de vidage d'un fichier principal qui est mémorisé dans un système de fichier ayant des éléments de mémoire à lecture-écriture (509) et à écriture unique-lecture multiple (519) pour mémoriser un contenu de fichier, le contenu du fichier comportant un ancien contenu (303, 305, 307) qui est mémorisé dans des éléments de mémoire à écriture unique-lecture multiple écrits et un nouveau contenu (311, 315) qui est mémorisé dans les éléments à lecture-écriture, le procédé comprenant les étapes de :

dans le cadre de l'opération de mémoire du nouveau contenu dans un élément de mémoire à lecture-écriture donné, établissement d'une correspondance entre l'élément à lecture-écriture donné et un élément de mémoire à écriture unique-lecture multiple non écrit donné ; et
en réponse à une commande de vidage, et à aucun autre moment, copie du contenu des éléments de mémoire à lecture-écriture dans les éléments de mémoire à écriture unique-lecture multiple non écrits correspondants (516) ;
l'étape de copie du nouveau contenu dans les éléments de mémoire à écriture unique-lecture multiple non écrits correspondants comporte les étapes de
marquage automatique de tous les éléments de mémoire à lecture-écriture qui contiennent le nouveau contenu ; et
copie non automatique du contenu des éléments de mémoire à lecture-écriture marqués dans leurs éléments de mémoire à écriture unique-lecture multiple non écrits correspondants.

5. Procédé selon la revendication 4 et comprenant en outre l'étape de :

création d'un fichier secondaire qui conserve l'état du fichier principal au moment de l'opération de vidage et dont le contenu est mémorisé dans les éléments de mémoire à écriture unique-lecture multiple écrits en réponse à la commande de vidage et dans les éléments de mémoire à écriture unique-lecture multiple contenant l'ancien contenu au moment de la commande de vidage.

6. Procédé selon la revendication 5, dans lequel :

un ancien contenu prédéterminé du fichier principal est mémorisé en plus dans les éléments de mémoire à lecture-écriture ;
l'étape de création du fichier secondaire comporte en outre l'étape de marquage automatique supplémentaire de tous les éléments de mémoire à lecture-écriture qui contiennent le nouveau contenu pour indiquer que les éléments de mémoire de lecture-écriture contiennent un ancien contenu.

FIG. 1

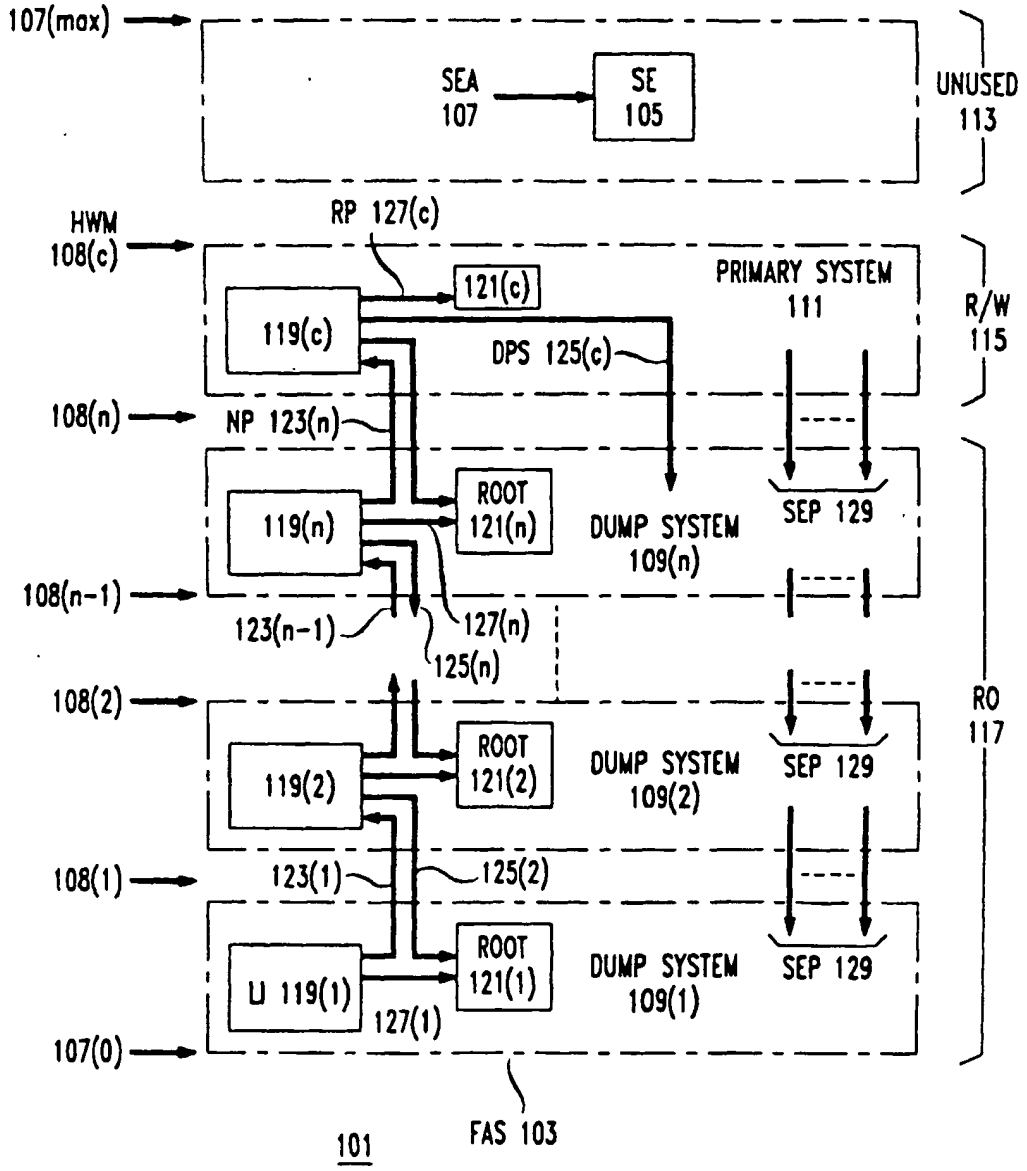

107(max)
SEA 107
SE 105
UNUSED 113
HWM 108(c)
RP 127(c)
121(c)
PRIMARY SYSTEM 111
R/W 115
119(c)
DPS 125(c)
108(n)
NP 123(n)
ROOT 121(n)
119(n)
DUMP SYSTEM 109(n)
SEP 129
108(n-1)
123(n-1)
127(n)
125(n)
RO 117
108(2)
ROOT 121(2)
119(2)
DUMP SYSTEM 109(2)
SEP 129
108(1)
123(1)
125(2)
ROOT 121(1)
LI 119(1)
DUMP SYSTEM 109(1)
SEP 129
127(1)
107(0)
FAS 103
101

FIG. 2

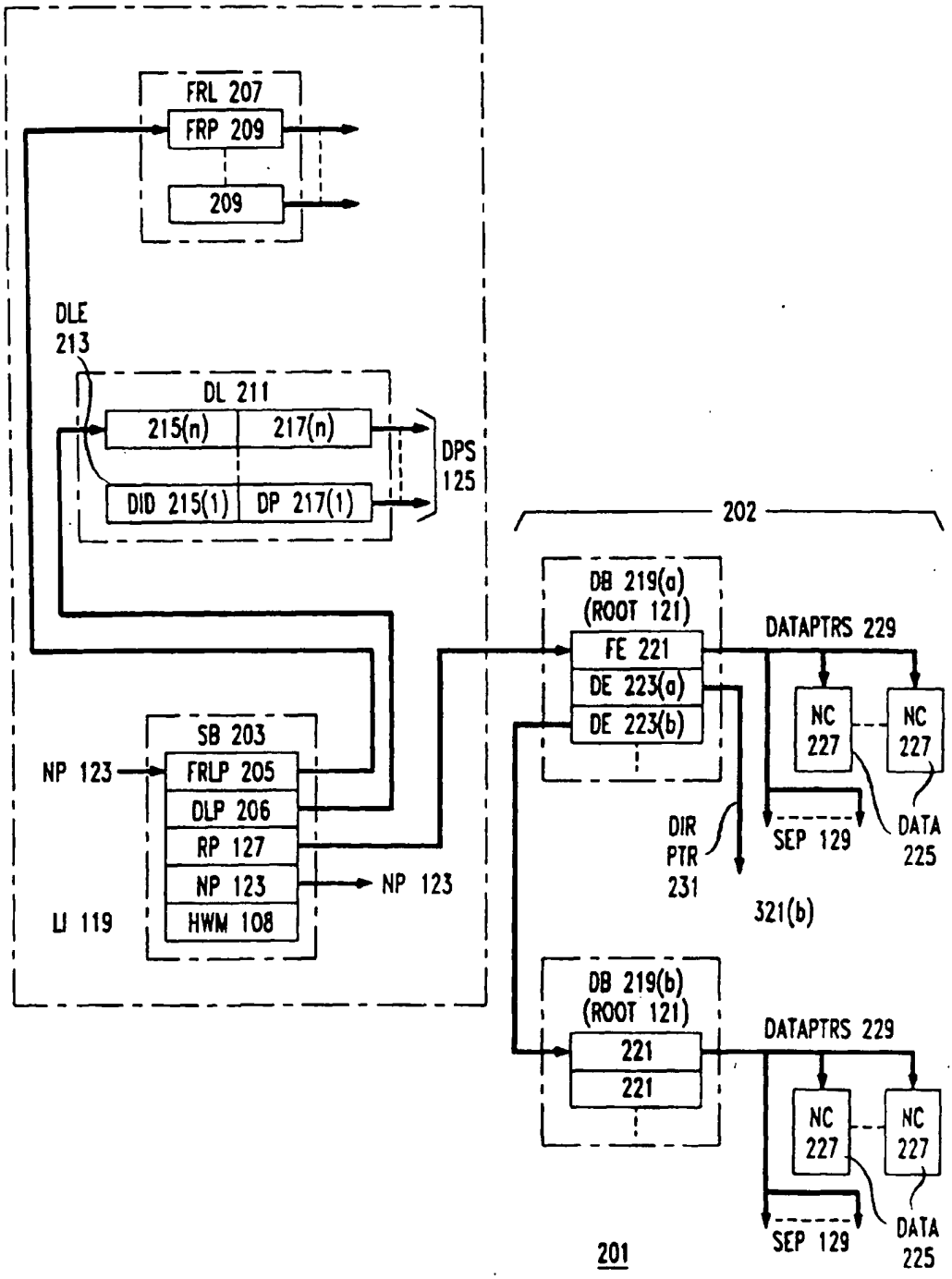

FRL 207
FRP 209
209
DLE 213
DL 211
215(n)
217(n)
DID 215(1)
DP 217(1)
DPS 125
202
DB 219(a) (ROOT 121)
FE 221
DE 223(a)
DE 223(b)
DATAPTRS 229
NC 227
NC 227
DATA 225
SEP 129
DIR PTR 231
321(b)
SB 203
NP 123
FRLP 205
DLP 206
RP 127
NP 123
NP 123
HWM 108
LI 119
DB 219(b) (ROOT 121)
221
221
DATAPTRS 229
NC 227
NC 227
DATA 225
SEP 129
201

FIG. 3

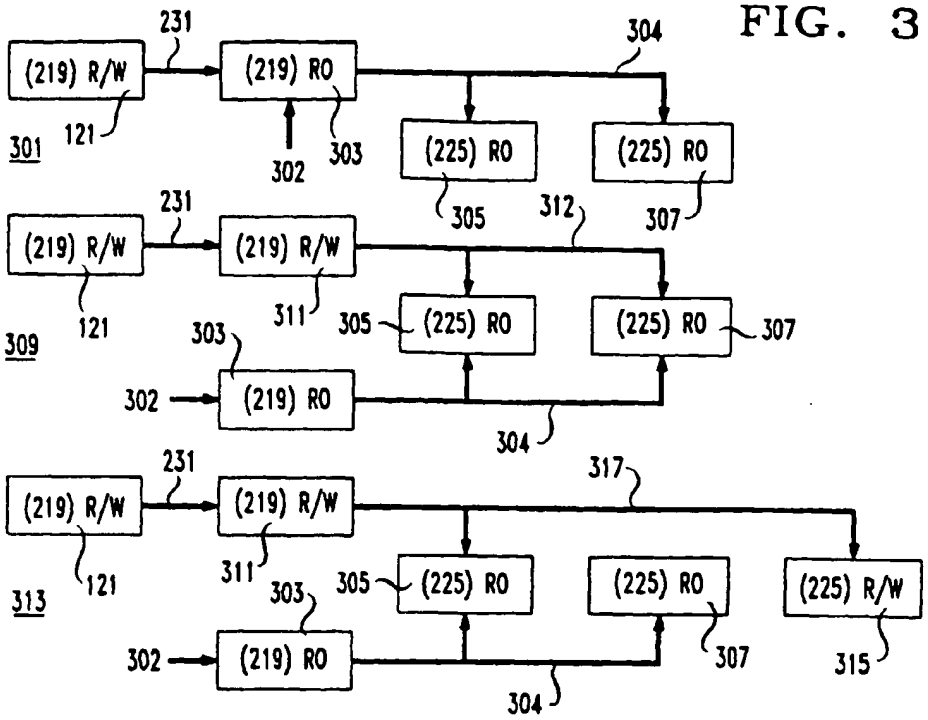

(219) R/W
231
(219) RO
304
301
121
303
302
(225) RO
(225) RO
305
312
307
(219) R/W
231
(219) R/W
309
121
311
305
(225) RO
(225) RO
307
303
302
(219) RO
304
(219) R/W
231
(219) R/W
317
313
121
311
303
305
(225) RO
(225) RO
(225) R/W
302
(219) RO
307
315
304

FIG. 4

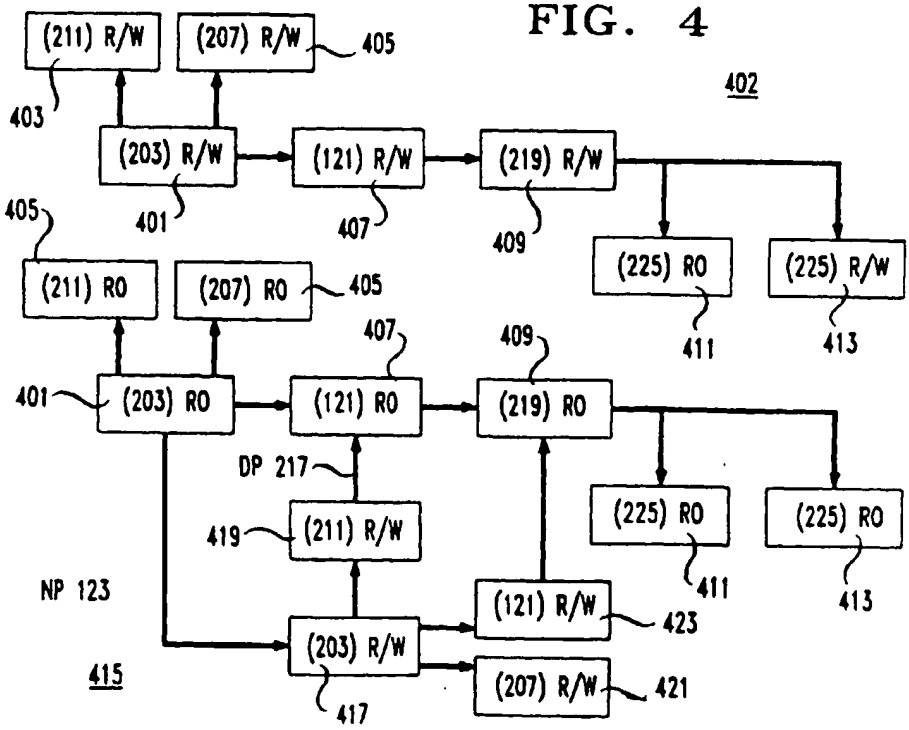

(211) R/W
(207) R/W
405
402
403
(203) R/W
(121) R/W
(219) R/W
405
401
407
409
(225) RO
(225) R/W
(211) RO
(207) RO
405
411
413
407
409
401
(203) RO
(121) RO
(219) RO
DP 217
(225) RO
(225) RO
419
(211) R/W
411
413
NP 123
(121) R/W
423
(203) R/W
415
(207) R/W
421
417

FIG. 5

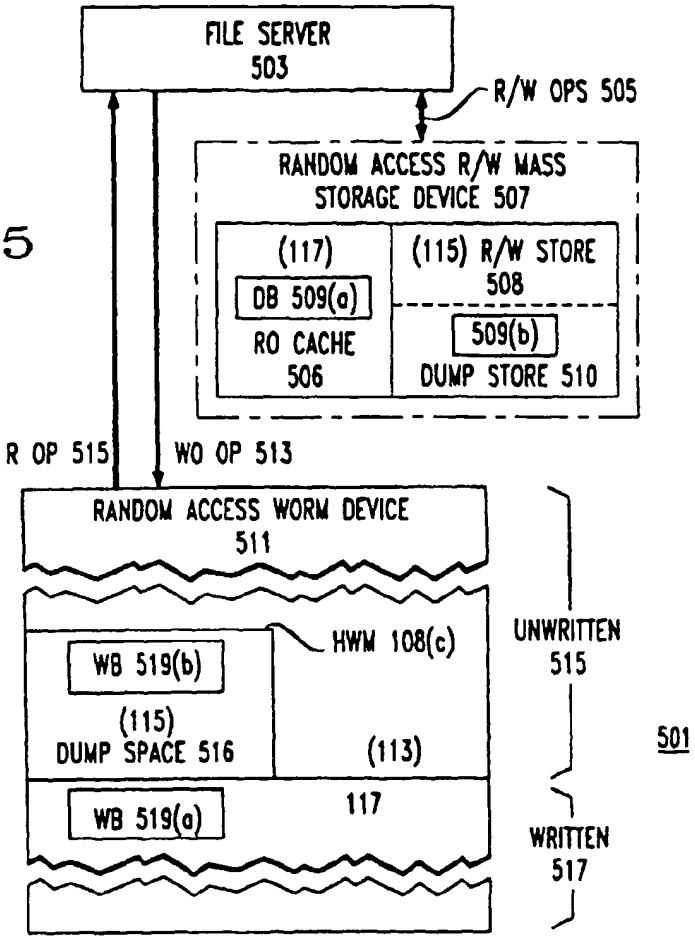

FILE SERVER
503
R/W OPS 505
RANDOM ACCESS R/W MASS
STORAGE DEVICE 507
(117)
DB 509(a)
RO CACHE
506
(115) R/W STORE
508
509(b)
DUMP STORE 510
R OP 515
WO OP 513
RANDOM ACCESS WORM DEVICE
511
HWM 108(c)
WB 519(b)
(115)
DUMP SPACE 516
(113)
WB 519(a)
117
UNWRITTEN
515
WRITTEN
517
501

FIG. 6

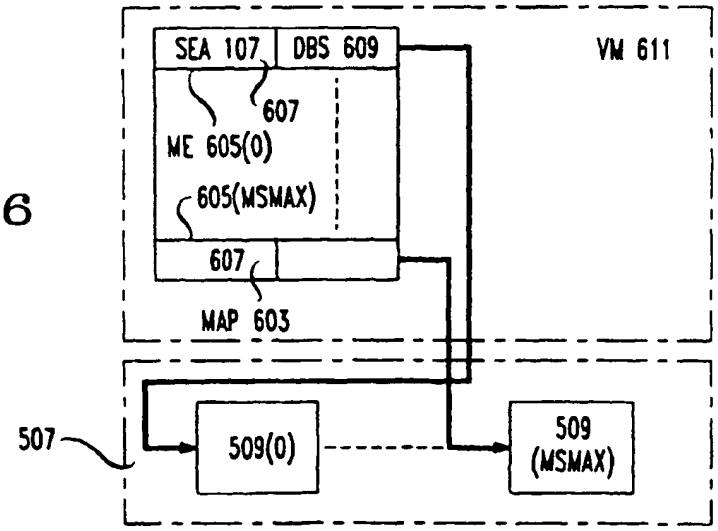
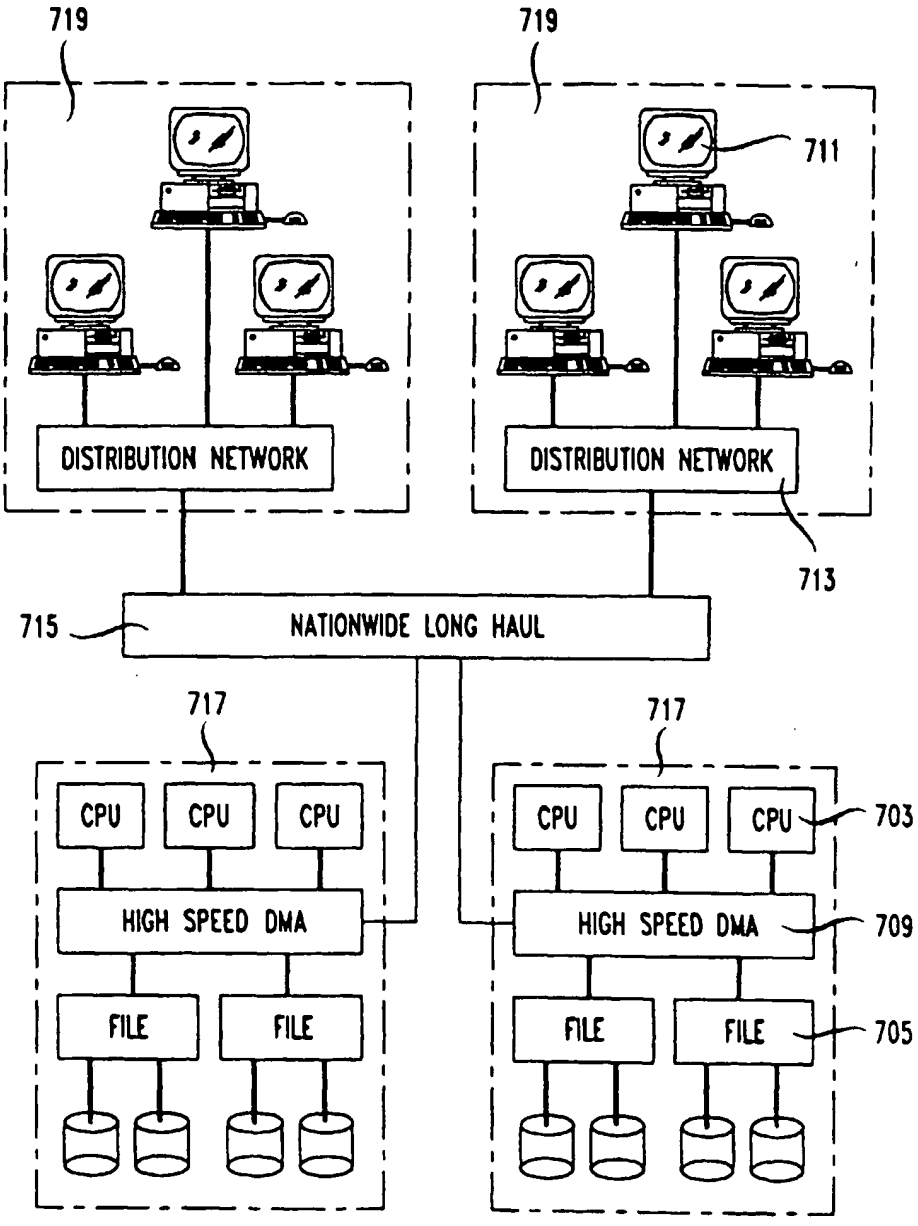

SEA 107
DBS 609
VM 611
607
ME 605(0)
605(MSMAX)
607
MAP 603
507
509(0)
509
(MSMAX)

FIG. 7


719
719
711
DISTRIBUTION NETWORK
DISTRIBUTION NETWORK
713
715
NATIONWIDE LONG HAUL
717
717
CPU
CPU
CPU
CPU
CPU
CPU
703
HIGH SPEED DMA
HIGH SPEED DMA
709
FILE
FILE
FILE
FILE
705
701